Towards Efficient Fragment of PDL

Mario Folhadela Benevides & Bruno Lopes & Leandro Gomes

Universidade Federal Fluminense
Niterói - RJ, Brasil

Second DL(R) Worshop - 2026

April 24, 2026

Outline

- Propositional Dynamic Logic - PDL
- Some classical extensions
  - Deterministic PDL - DPDL
  - PDL with **while** programs - SPDL
- Kleen Algebras with Test - KAT
- Guarded Kleen Algebras with Test - GKAT
- Guarded Propositional Dynamic Logic - GPDL
- Final Remarks

Dynamic Logics

- Three Broad Categories:

Dynamic Logics

- Program specification;
- Modalities are programs

Dynamic Logics: Action Logics

- AI: planning and reasoning about actions;
- Modalities are actions performed by agents;

Dynamic Epistemic Logics

- Reasoning about knowledge;
- Modalities are epistemic actions performed by agents;
- actions that change agents knowledge.

Dynamic Logics

Program Verification

Engeles (67), Floyd (67), Hoare (69), Manna (74)

Modal Approach

Salwicki (70) - Algorithm Logic
Pratt (76) - Dynamic Logic

Why Modal Logic?

- Two interesting problems:
  - Satisfaction: Given a formula ϕ and a structure $\mathcal{E}$, we want to know if ϕ holds at $\mathcal{E}$.
  - Validity/Satisfiability: Given a formula ϕ, we want to know if there exists a structure $\mathcal{E}$ where ϕ holds.
- Over finite structures
  - Model Checking: Given a formula ϕ and a finite structure $\mathcal{E}$, we want to know if ϕ holds at $\mathcal{E}$.
  - Validity: Given a formula ϕ, we want to know if there exists a finite structure $\mathcal{E}$ where ϕ holds.

F.O.L.

- Satisfaction: Given a ϕ and a structure $\mathcal{E}$, we want to know if ϕ holds in $\mathcal{E}$.

  Undecidable

- Validity: Given a ϕ, we want to know if there exists a structure $\mathcal{E}$ where ϕ holds.

  Undecidable

Over finite structures

- Model Checking: Given a ϕ and a finite structure $\mathcal{E}$, we want to know if ϕ holds in $\mathcal{E}$.

  PSPACE

- Validity: Given a ϕ, we want to know if there exists a finite structure $\mathcal{E}$ where ϕ holds.

  Undecidable

Modal Logics (in general)

- Satisfaction: Given a ϕ and a structure $\mathcal{E}$, we want to know if ϕ holds in $\mathcal{E}$.

  Decidable

- Validity: Given a ϕ, we want to know if there exists a structure $\mathcal{E}$ where ϕ holds.

  Decidable

Over finite structures

- Model Checking: Given a ϕ and a finite structure $\mathcal{E}$, we want to know if ϕ holds in $\mathcal{E}$.

  LINEAR

- Validity: Given a ϕ, we want to know if there exists a finite structure $\mathcal{E}$ where ϕ holds.

  Decidable

Complexity - Modal Logics

- Model Checking: $O(|\varphi| \times (|W| + |R|))$ LINEAR
- Validity: **K**, **T** e **S4** is **PSPACE**-Complete.
- Validity: **S5** is **NP**-Complete.
- **P** $\subseteq$ **NP** $\subseteq$ **PSPACE** $\subseteq$ **EXPTIME**
- Validity: some are **EXPTIME**-Complete,
- **In General**:
  - **Decidable**
  - **Model Checking**: LINEAR

Historical Remarks

- Vaughan Pratt (mid-70s)
- Fisher & Ladner (1976)
  - Finite Model Property;
  - Decidability;
- Segerberg (1977)
  - 1^{st} Proof of Completeness (contained a gap);
  - Induction axioms (Ancestral Logic)
- Rohit Parikh (1978)
  - 1^{st} Correct Proof of Completeness;
- Parikh and Kozen (1981)
  - Shorter Proof of Completeness;

Dynamic Logics

Main Features

- Actions
- Programs
- Induction axiom
- Fixed Point operator:
  - reflexive and transitive closure
  - least/greatest fixed point
  - finite paths/executions

Dynamic Logics

Motivation

- Reasoning about Programs
- Correctness of Programs
- Equivalence of Programs
- Expressive Power of Program Constructors
- Synthesis of Program from specification

Propositional Dynamic Logic - PDL

PDL

- Propositional version o D. L.
- Reasoning about programs and propositions which are independent of the domain
- No assignment to variables
- multi-modal logic with one modality $\langle \pi \rangle$ for each program π
- Used in formal specification and to reason about programs and actions.

PDL for Regular Programs

Regular PDL

Prop. Modal Logic

$+$

Alg. of Regular Expressions

Language

- multi-modal logic with one modality $\langle \pi \rangle$ for each program π

PDL for Regular Programs

Regular PDL

Prop. Modal Logic

$+$

Alg. of Regular Expressions

Language

- multi-modal logic with one modality $\langle \pi \rangle$ for each program π
- basic programs

PDL for Regular Programs

Regular PDL

Prop. Modal Logic

$+$

Alg. of Regular Expressions

Language

- multi-modal logic with one modality $\langle \pi \rangle$ for each program π
- basic programs
- sequential composition **;**
- non-deterministic choice $\cup$
- Iteration $\star$
- test **?**

PDL Language

- Formulas

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle\pi\rangle\varphi \mid [\pi]\varphi$$

- Programs

$$\pi ::= a \mid \pi_1; \pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \mid \varphi?$$

- $\langle\pi\rangle\varphi$: it is possible to execute π and it halts in a state satisfying φ
- $[\pi]\varphi$: whenever π halts, it does in a state satisfying φ

More PDL Operators

- if A then α else $\beta \Rightarrow (A?;\alpha) \cup (\neg A;\alpha)$
- - $\alpha^+ \Rightarrow \alpha;\alpha^\star$
- - while A do $\alpha \Rightarrow (A?;\alpha)^\star;\neg A?$
- - repeat α until $\neg A \Rightarrow (\alpha;((A?;\alpha)^\star;\neg A?))$
- - abort $\Rightarrow 0?$
- - skip $\Rightarrow 1?$

Semantics

- A *model* for PDL is a tuple $\mathcal{M} = (W, R_a, V)$ where
  - W is a non-empty set of states;
  - R_a is a binary relation for each basic program a;
  - R_π is a binary relation for each non-basic program π,
    - $R_{\pi_1;\pi_2} = R_{\pi_1} \circ R_{\pi_2}$,
    - $R_{\pi_1 \cup \pi_2} = R_{\pi_1} \cup R_{\pi_2}$,
    - $R_{\varphi?} = \{(w, w) \mid \mathcal{M}, w \Vdash \varphi\}$,
    - $R_{\pi^\star} = R^\star_\pi$, where $R^\star_\pi$ is the reflexive transitive closure of R_π.
  - V is a valuation function $\mathbf{V} : \Phi \mapsto 2^W$
- We say that $\mathcal{F} = (W, \mathbf{R})$ is a PDL frame.

PDL - Satisfaction

Satisfaction

- $\mathcal{M}, w \Vdash p$ iff $w \in \mathbf{V}(p)$;
- $\mathcal{M}, w \Vdash \top$ always;
- $\mathcal{M}, w \Vdash \neg\varphi$ iff $\mathcal{M}, w \nVdash \varphi$;
- $\mathcal{M}, w \Vdash \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, w \Vdash \varphi_1$ and $\mathcal{M}, w \Vdash \varphi_2$;
- $\mathcal{M}, w \Vdash \langle\pi\rangle\varphi$ iff there is w', $wR_\pi w'$ and $\mathcal{M}, w' \Vdash \varphi$.

PDL+ Axiomatization

Axioms

1. *All tautologies*,
2. $[\pi](\varphi \to \psi) \to ([\pi]\varphi \to [\pi]\psi)$,
3. $[\pi_1;\pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi$,
4. $[\pi_1 \cup \pi_2]\varphi \leftrightarrow [\pi_1]\varphi \wedge [\pi_2]\varphi$,
5. $[\pi^*]\varphi \leftrightarrow \varphi \wedge [\pi][\pi^*]\varphi$,
6. $[\pi^*](\varphi \to [\pi]\varphi) \to (\varphi \to [\pi^*]\varphi)$,
7. $[\varphi?]\psi \leftrightarrow \varphi \wedge \psi$

Inference Rules

M.P. $\varphi, \varphi \to \psi / \psi$ U.G. $\varphi / [\pi]\varphi$ SUB. $\varphi / \sigma\varphi$

Soundness, Completeness & Complexity

- **Soundness**: PDL is sound.
- **Completeness**: PDL is complete with respect to the class of finite PDL models.
- **Compactness**: PDL is not compact.
- **Decidability**: PDL is decidable and has f.m.p..
- **Complexity - Validity/Satisfiabilty**: EXPTIME-complete.
- **Complexity Model Checking**: PTIME.

Some Classical Extensions of PDL

Extensions/Variants

- Deterministic PDL
- PDL with Converse
- Restricting Test
- Context-Free Programs
- Nominals and Binders (Hybrid)

Some Classical Extensions of PDL

Deterministic PDL - DPDL

- Language: same
- Semantics: R_a are partial functions, a basic program
- Axiom: $\langle a \rangle p \rightarrow [a]p$
- Decidable: f.m.p. & filtration
- Sat. $\Rightarrow$ EXPTIME-complete
- M.C. $\Rightarrow$ PTIME-complete
- DPDL $\subset$ PDL

Condition 1:

Only deterministic basic programs

Strict PDL

SPDL

- Restrict the use of the operators $\cup$ and *.
- Only deterministic while programs are allowed;
- Include the following deterministic programs.

  if φ **then** α **else** β $=$ $\varphi?;\alpha \cup \neg\varphi?;\beta$

  while φ **do** α $=$ $(\varphi?;\alpha)^\star;\neg\varphi?$
- Finitely Axiomimatizable
- Decidable: f.m.p. & filtration
- Sat. $\Rightarrow$ EXPTIME-complete
- SPDL $\subset$ PDL

Condition 2:

Only deterministic while programs are allowed

Strict Deterministic PDL

SDPDL

We have both conditions: DPDL and SPDL restrictions.

- **Condition 1:** Only deterministic basic programs
- **Condition 2:** Only deterministic while programs are allowed
- Finitely Axiomimatizable
- Decidable: f.m.p. & filtration
- Validity $\Rightarrow$ PSPACE-complete ☺
- SDPDL $\subset$ DPDL and SPDL $\subset$ PDL

Validity complexity

PDL: EXPTime-Complete

Validity complexity

PDL: EXPTime-Complete

DPDL: EXPTime-Complete

Validity complexity

PDL: EXPTime-Complete

DPDL: EXPTime-Complete

SPDL: EXPTime-Complete

Validity complexity

PDL: EXPTime-Complete

DPDL: EXPTime-Complete

SPDL: EXPTime-Complete

SDPDL: PSPACE-complete ☺

Guarded Propositional Dynamic Logic - GPDL

- GPDL is quite similar to SDPDL
- **Condition 1:** Only deterministic basic programs
- **Condition 2:** Only deterministic while programs are allowed
- but we restrict *test* operator ? to booleans b, where b is a boolean dormula
- GKAT is the algebraic counter part of GPDL

KAT

Definition

A Kleene algebra with tests is a structure

$$(K, B, +, \cdot, ^\star, 0, 1, \neg)$$

where $B \subseteq K$, $(K, +, \cdot, ^\star, 0, 1)$ is a Kleene algebra and $(B, +, \cdot, 0, 1, \neg)$ is a Boolean subalgebra. The elements of K are called programs and elements of B tests.

- a set of binary relations over a set W,
- operators: of union ($\cup$), relational composition ($\circ$), the reflexive and transitive closure (*), the empty relation ($\varnothing$) and the identity relation (Δ), embedded with a Booleean subalgebra for tests.
- Tests are presented as subsets of the identity relation and the negation operator on tests is given as the complement of those subsets.
- KAT is the algebraic counterpart of PDL.

GKAT

- GKAT Guarded Kleene algebra with tests is the algebraic counterpart of GPDL.
- the terms of GKAT will constitute the set of GPDL programs.
- GKAT is less expressive than KAT, since it does not allow for arbitrary use of nondeterminism and iteration;
- still makes possible the encoding of imperative commands **if-then-else** and **while-do** by embedding the element of the Boolean subalgebra into the operators.
- those commands are translated as $(\textbf{if } b \textbf{ then } \pi_1 \textbf{ else } \pi_2) \equiv \pi_1 +_b \pi_2$ and $(\textbf{while } b \textbf{ do } \pi) \equiv \pi^{(b)}$.

GKAT

- GKAT is less expressive than KAT, since it does not allow for arbitrary use of nondeterminism and iteration;
- (**if** b **then** π_1 **else** π_2) $\equiv \pi_1 +_b \pi_2$ and (**while** b **do** π) $\equiv \pi^{(b)}$.
- **iteration** and **non-determinism** in a controlled way;
- KAT: complexity in verifying the equivalence of two programs is PSPACE-Complete;
- GKAT: *the equivalence of GKAT porgrams e and f can be established in almost linear time* $O(n \times \alpha(n))$, where α denotes the inverse Ackermann function and $n = (\mid e \mid + \mid f \mid)$.
- The algorithm consists in converting the expressions in Thompson automatas (which computes in $O(n)$), normalize them (which computes in $O(n)$), and check bisimilarity by an adapted version of Hopcroft-Karp algorithm (which computes in $O(n \times \alpha(n))$).

GKAT

- **iteration** and **non-determinism** in a controlled way;
- KAT: complexity in verifying the equivalence of two programs is PSPACE-Complete;
- GKAT: *the equivalence of GKAT porgrams e and f can be established in almost linear time*
- **Complexity**: $O(n \times \alpha(n))$, where α denotes the inverse Ackermann function and $n = (\mid e \mid + \mid f \mid)$.

Guarded Propositional Dynamic Logic - GPDL

- GPDL is quite similar to SDPDL
- **Condition 1:** Only deterministic basic programs
- **Condition 2:** Only deterministic while programs are allowed
- but we restrict *test* ? operator to booleans b, where b is a boolean formula
- GKAT is the algebraic counter part of GPDL
- Since GPDL programs are built using the same syntax as GKAT terms;
- GPDL: *the equivalence of GPDL porgrams e and f can be established in almost linear time* $O(n \times \alpha(n))$, where α denotes the inverse Ackermann function and $n = (\mid e \mid + \mid f \mid)$.

Guarded Propositional Dynamic Logic - GPDL

- GPDL programs are built using the same syntax as GKAT terms,
- to use the algorithm to compute equivalence between GPDL programs.

Theorem:

• two GKAT terms are equivalent if and only if the two correspondent GPDL programs are logically equivalent.

$$[\![e]\!] = [\![f]\!] \Leftrightarrow \ \models \langle e \rangle p \leftrightarrow \langle f \rangle p$$

Main Result:

• GPDL: *the equivalence of GPDL porgrams e and f can be established in almost linear time*

• $O(n \times \alpha(n))$, where α denotes the inverse Ackermann function and $n = (\mid e \mid + \mid f \mid)$.

Guarded Propositional Dynamic Logic - GPDL

GPDL

We have both conditions: DPDL and SPDL restrictions.

- **Condition 1:** Only deterministic basic programs
- **Condition 2:** Only deterministic while programs are allowed
- but we restrict *test* ? operator to booleans b, where b is a boolean formula
- Finitely Axiomimatizable
- Decidable: f.m.p. & filtration
- Sat. $\Rightarrow$ almost linear ☺
- GPDL $\subset$ DPDL and SPDL $\subset$ PDL

Validity complexity

PDL: EXPTime-Complete

Validity complexity

PDL: EXPTime-Complete

DPDL: EXPTime-Complete

Validity complexity

PDL: EXPTime-Complete

DPDL: EXPTime-Complete

SPDL: EXPTime-Complete

Validity complexity

PDL: EXPTime-Complete

DPDL: EXPTime-Complete

SPDL: EXPTime-Complete

SDPDL: PSPACE-complete

Validity complexity

PDL: EXPTime-Complete

DPDL: EXPTime-Complete

SPDL: EXPTime-Complete

SDPDL: PSPACE-complete

GPDL: Almost Linear ☺

Validity complexity

PDL: EXPTime-Complete

DPDL: EXPTime-Complete

SPDL: EXPTime-Complete

SDPDL: PSPACE-complete

GPDL: Almost Linear ☺

Validity complexity

PDL: EXPTime-Complete

DPDL: EXPTime-Complete

SPDL: EXPTime-Complete

SDPDL: PSPACE-complete

GPDL: Almost Linear ☺ for equivalence of Programs

Final Remarks

Guarded Propositional Dynamic Logic - GPDL

- Finitely Axiomimatizable
- Decidable: f.m.p. & filtration
- **Validity**: $\Rightarrow$ almost linear ☺

Final Remarks

Guarded Propositional Dynamic Logic - GPDL

- Finitely Axiomimatizable
- Decidable: f.m.p. & filtration
- **Validity**: $\Rightarrow$ almost linear ☺ for equivalence of Programs
- $[\![e]\!] = [\![f]\!] \Leftrightarrow \models \langle e \rangle p \leftrightarrow \langle f \rangle p$.
- *the equivalence of GPDL porgrams e and f can be established in almost linear time* $O(n \times \alpha(n))$, where α denotes the inverse Ackermann function and $n = (\mid e \mid + \mid f \mid)$.
- GPDL $\subset$ DPDL and SPDL $\subset$ PDL

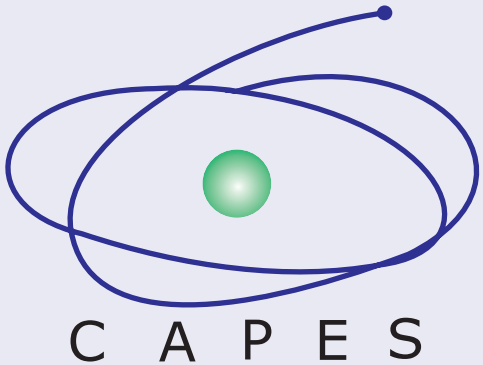
Future

Efficient Fragment of PDL

- to implement a program to verify the equivalence of GPDL programs based on the algorithm proposed;
- to use the same technique to investigate other Dynamic Logics;
- to investigate proof systems for fragments of PDL with better complexity:
  - Tableaux
  - Natural Deduction
  - Seguent Calculus

Thank you!


CNPq


CAPES


FAPERJ
Fundação Carlos Chagas Filho de Amparo
à Pesquisa do Estado do Rio de Janeiro

Contact information

Mario Benevides (mario@ic.uff.br)
Bruno Lopes (bruno@ic.uff.br)

http://www.ic.uff.br/∼mario
http://www.ic.uff.br/∼bruno
Instituto de Computação - Universidade Federal Fluminense - Niterói-RJ,